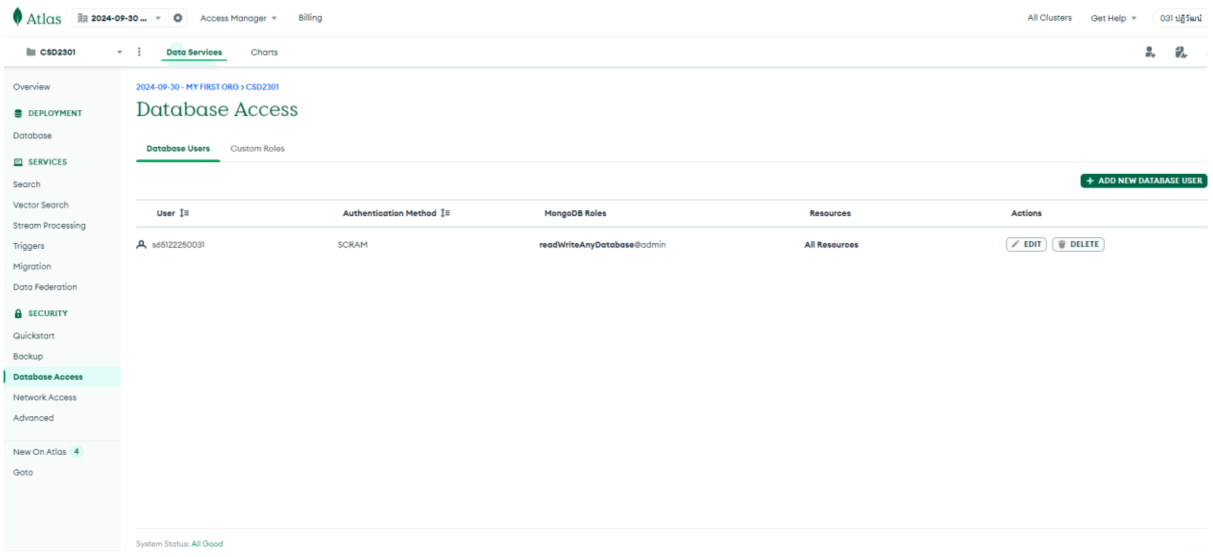
3.1

1.



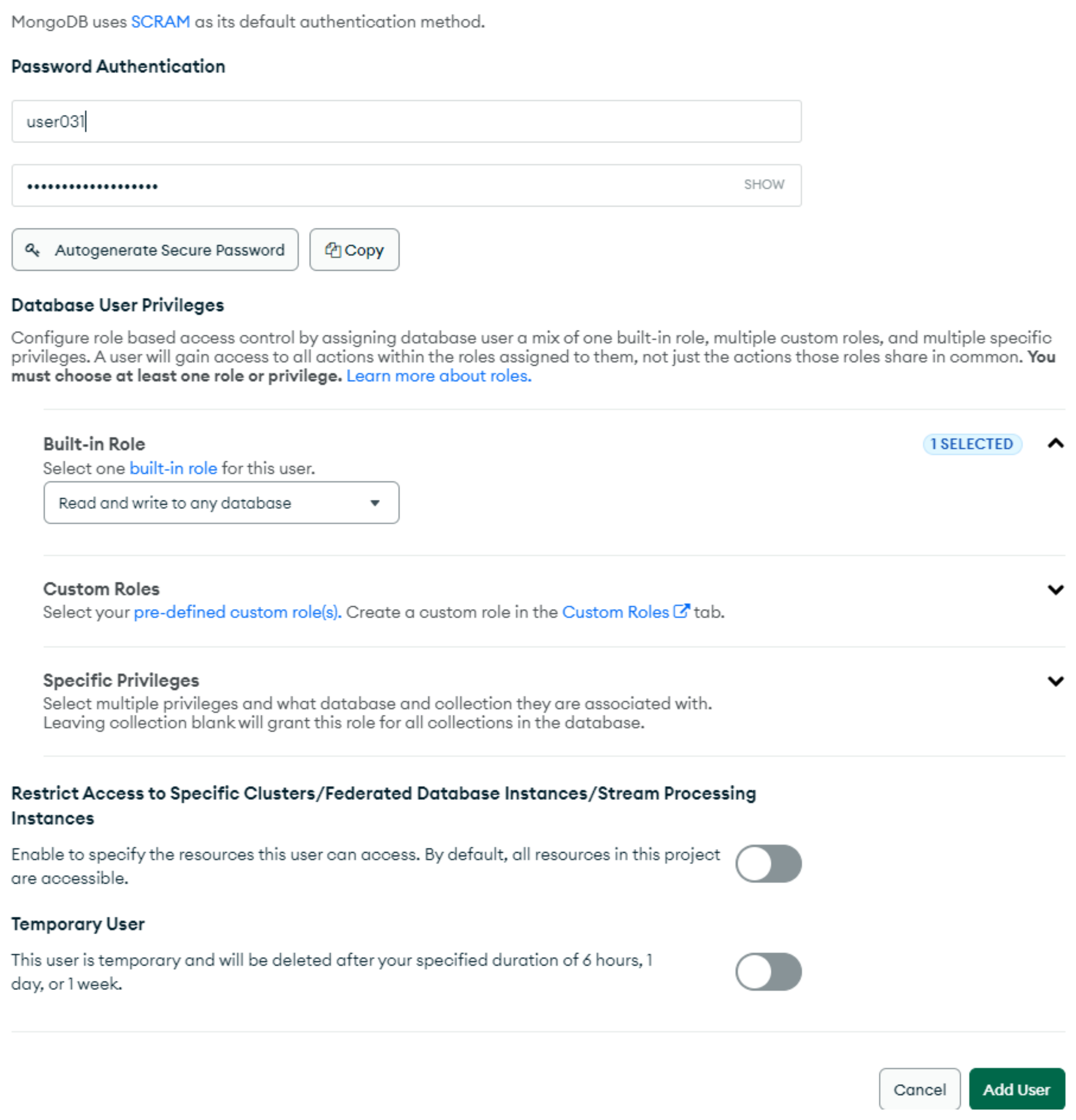
2024-09-30 - MY FIRST ORG › CSD2301

Database Access

User	Authentication Method	MongoDB Roles	Resources	Actions
s65122250031	SCRAM	**readWriteAnyDatabase**@admin	**All Resources**	EDIT DELETE

System Status: All Good

2.


MongoDB uses SCRAM as its default authentication method.

Password Authentication

user031

•••••••••••••••••• SHOW

Autogenerate Secure Password | Copy

Database User Privileges

Configure role based access control by assigning database user a mix of one built-in role, multiple custom roles, and multiple specific privileges. A user will gain access to all actions within the roles assigned to them, not just the actions those roles share in common. **You must choose at least one role or privilege.** Learn more about roles.

Built-in Role 1 SELECTED

Select one built-in role for this user.

Read and write to any database ▾

Custom Roles

Select your pre-defined custom role(s). Create a custom role in the Custom Roles tab.

Specific Privileges

Select multiple privileges and what database and collection they are associated with.
Leaving collection blank will grant this role for all collections in the database.

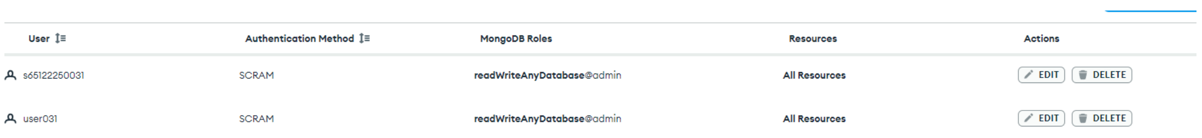
Restrict Access to Specific Clusters/Federated Database Instances/Stream Processing Instances

Enable to specify the resources this user can access. By default, all resources in this project are accessible.

Temporary User

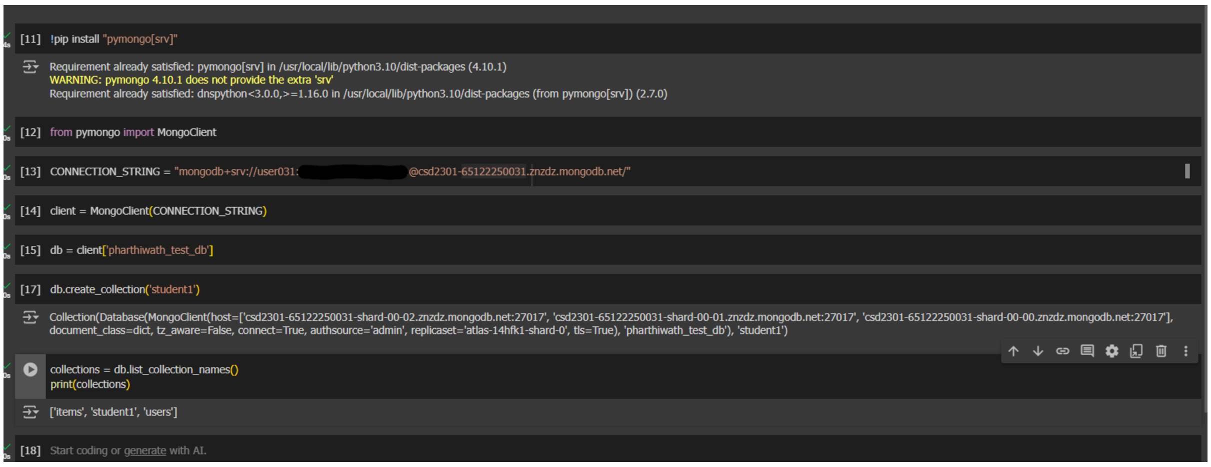
This user is temporary and will be deleted after your specified duration of 6 hours, 1 day, or 1 week.

Cancel | Add User



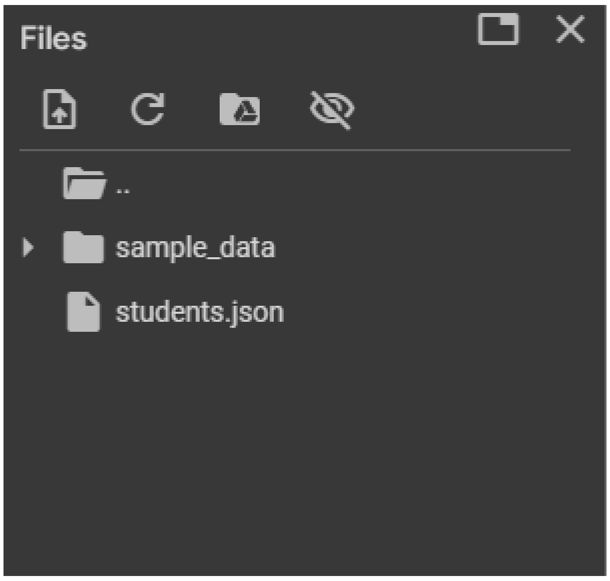
User	Authentication Method	MongoDB Roles	Resources	Actions
s65122250031	SCRAM	**readWriteAnyDatabase**@admin	**All Resources**	EDIT DELETE
user031	SCRAM	**readWriteAnyDatabase**@admin	**All Resources**	EDIT DELETE

3.

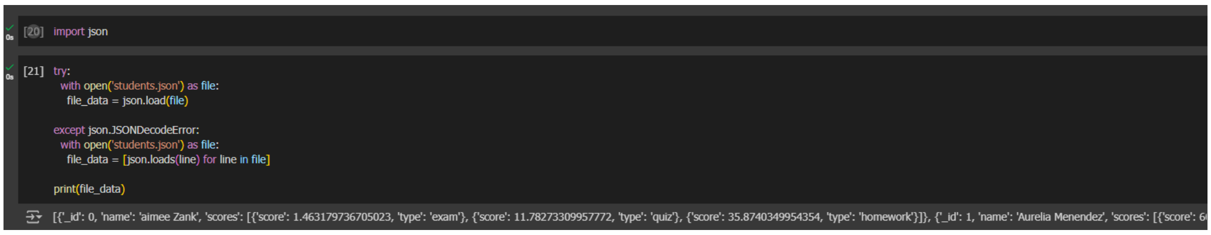


```
[11] !pip install "pymongo[srv]"
Requirement already satisfied: pymongo[srv] in /usr/local/lib/python3.10/dist-packages (4.10.1)
WARNING: pymongo 4.10.1 does not provide the extra 'srv'
Requirement already satisfied: dnspython<3.0.0,>=1.16.0 in /usr/local/lib/python3.10/dist-packages (from pymongo[srv]) (2.7.0)

[12] from pymongo import MongoClient

[13] CONNECTION_STRING = "mongodb+srv://user031:[redacted]@csd2301-65122250031.znzdz.mongodb.net/"

[14] client = MongoClient(CONNECTION_STRING)

[15] db = client['pharthiwath_test_db']

[17] db.create_collection('student1')
Collection(Database(MongoClient(host=['csd2301-65122250031-shard-00-02.znzdz.mongodb.net:27017', 'csd2301-65122250031-shard-00-01.znzdz.mongodb.net:27017', 'csd2301-65122250031-shard-00-00.znzdz.mongodb.net:27017'],
document_class=dict, tz_aware=False, connect=True, authsource='admin', replicaset='atlas-14hfk1-shard-0', tls=True), 'pharthiwath_test_db'), 'student1')

collections = db.list_collection_names()
print(collections)
['items', 'student1', 'users']

[18] Start coding or generate with AI.
```

5.

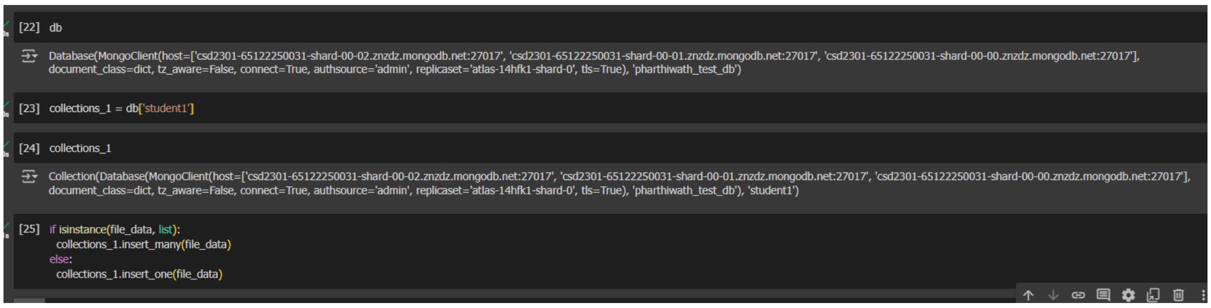


6.

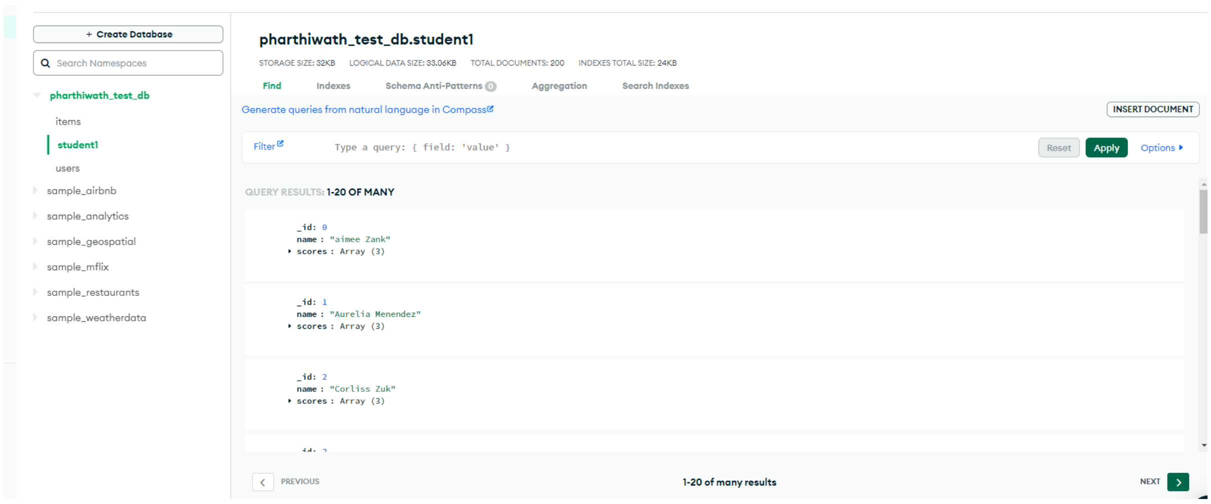


```
[20] import json

[21] try:
       with open('students.json') as file:
         file_data = json.load(file)

     except json.JSONDecodeError:
       with open('students.json') as file:
         file_data = [json.loads(line) for line in file]

     print(file_data)

[{'_id': 0, 'name': 'aimee Zank', 'scores': [{'score': 1.463179736705023, 'type': 'exam'}, {'score': 11.78273309957772, 'type': 'quiz'}, {'score': 35.8740349954354, 'type': 'homework'}]}, {'_id': 1, 'name': 'Aurelia Menendez', 'scores': [{'score': 6
```

7.



```
[22] db

Database(MongoClient(host=['csd2301-65122250031-shard-00-02.znzdz.mongodb.net:27017', 'csd2301-65122250031-shard-00-01.znzdz.mongodb.net:27017', 'csd2301-65122250031-shard-00-00.znzdz.mongodb.net:27017'],
document_class=dict, tz_aware=False, connect=True, authsource='admin', replicaset='atlas-14hfk1-shard-0', tls=True), 'pharthiwath_test_db')

[23] collections_1 = db['student1']

[24] collections_1

Collection(Database(MongoClient(host=['csd2301-65122250031-shard-00-02.znzdz.mongodb.net:27017', 'csd2301-65122250031-shard-00-01.znzdz.mongodb.net:27017', 'csd2301-65122250031-shard-00-00.znzdz.mongodb.net:27017'],
document_class=dict, tz_aware=False, connect=True, authsource='admin', replicaset='atlas-14hfk1-shard-0', tls=True), 'pharthiwath_test_db'), 'student1')

[25] if isinstance(file_data, list):
       collections_1.insert_many(file_data)
     else:
       collections_1.insert_one(file_data)
```



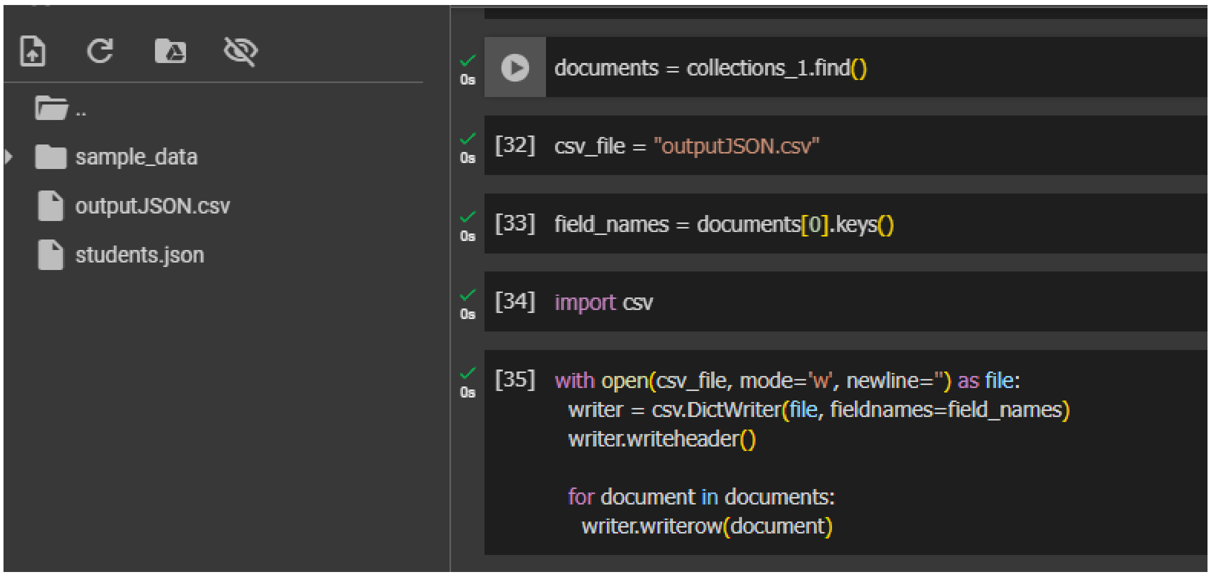
pharthiwath_test_db.student1

STORAGE SIZE: 32KB LOGICAL DATA SIZE: 33.06KB TOTAL DOCUMENTS: 200 INDEXES TOTAL SIZE: 24KB

QUERY RESULTS: 1-20 OF MANY

```
_id: 0
name : "aimee Zank"
scores : Array (3)

_id: 1
name : "Aurelia Menendez"
scores : Array (3)

_id: 2
name : "Corliss Zuk"
scores : Array (3)
```

8.



```
documents = collections_1.find()
```

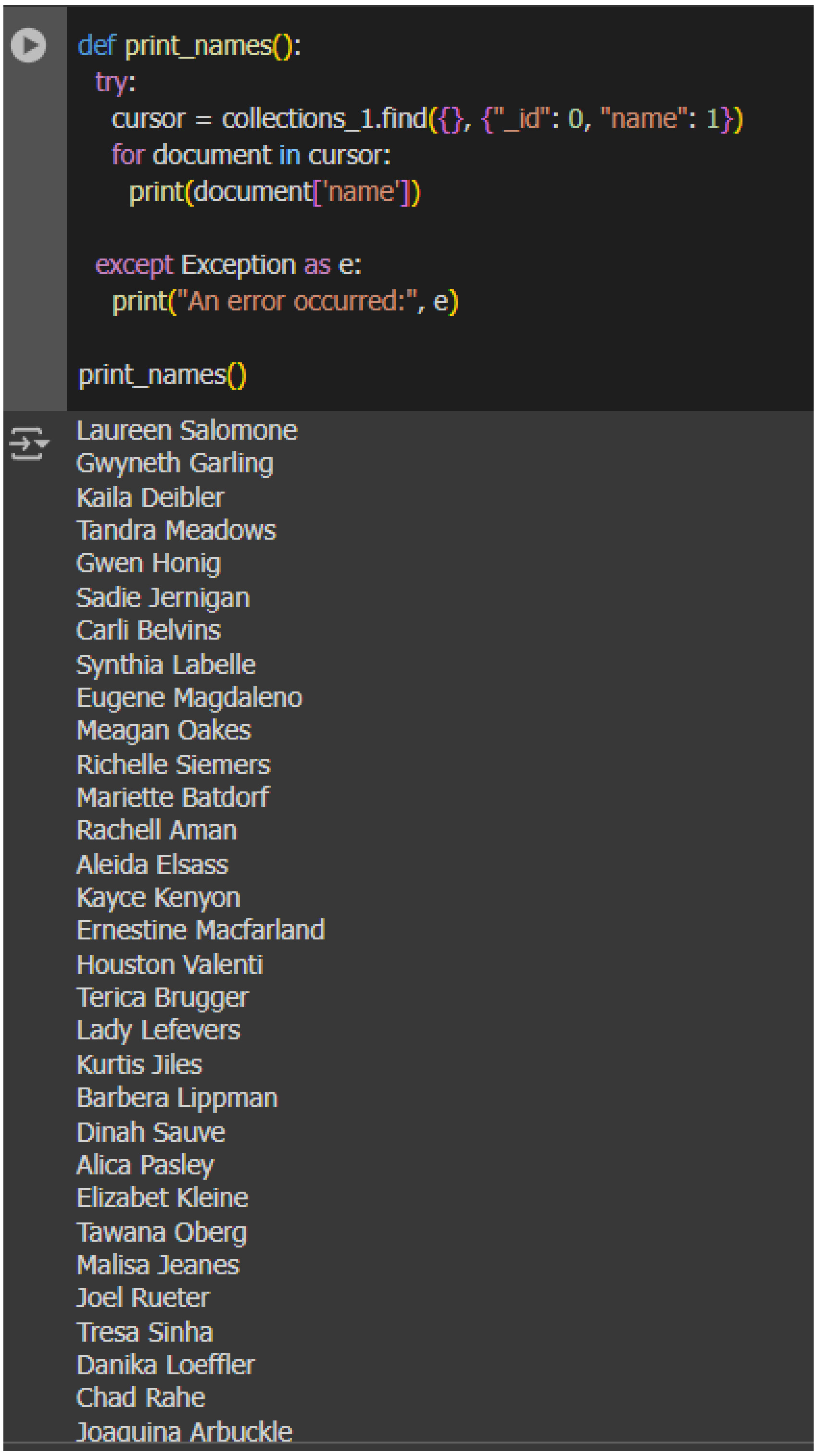
```
[32] csv_file = "outputJSON.csv"
```

```
[33] field_names = documents[0].keys()
```

```
[34] import csv
```

```
[35] with open(csv_file, mode='w', newline='') as file:
       writer = csv.DictWriter(file, fieldnames=field_names)
       writer.writeheader()

       for document in documents:
         writer.writerow(document)
```

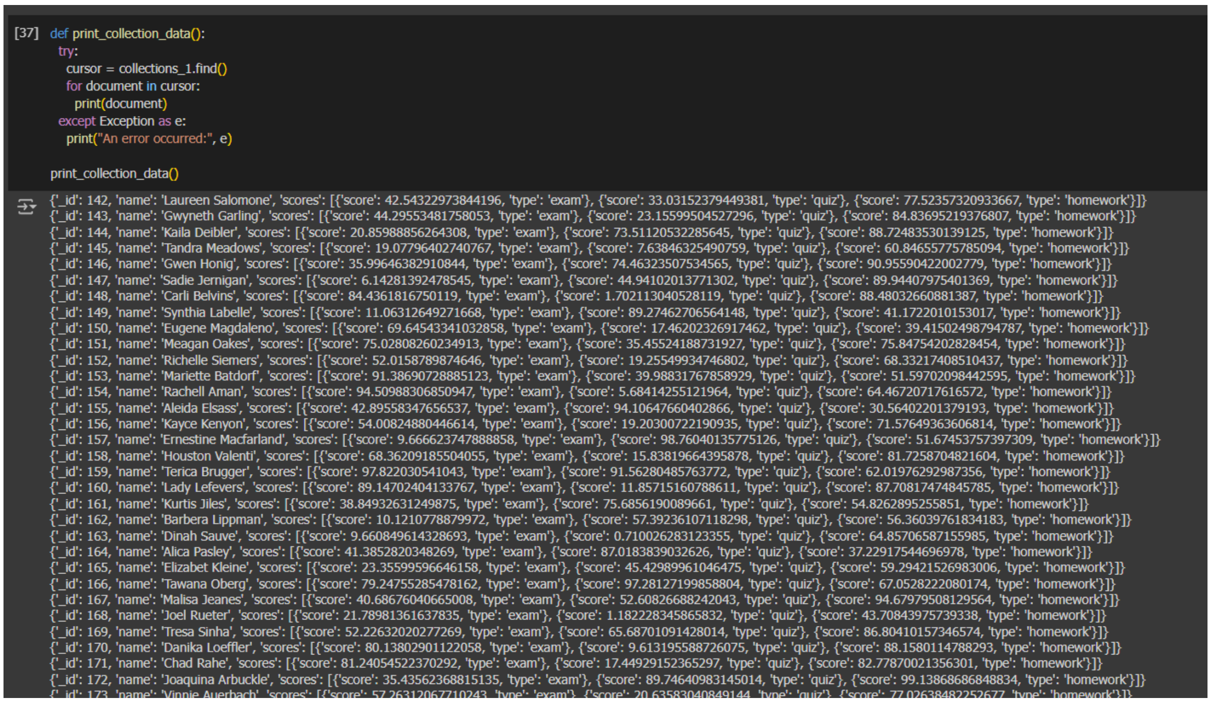
9.



```
def print_names():
  try:
    cursor = collections_1.find({}, {"_id": 0, "name": 1})
    for document in cursor:
      print(document['name'])

  except Exception as e:
    print("An error occurred:", e)

print_names()
```

```
Laureen Salomone
Gwyneth Garling
Kaila Deibler
Tandra Meadows
Gwen Honig
Sadie Jernigan
Carli Belvins
Synthia Labelle
Eugene Magdaleno
Meagan Oakes
Richelle Siemers
Mariette Batdorf
Rachell Aman
Aleida Elsass
Kayce Kenyon
Ernestine Macfarland
Houston Valenti
Terica Brugger
Lady Lefevers
Kurtis Jiles
Barbera Lippman
Dinah Sauve
Alica Pasley
Elizabet Kleine
Tawana Oberg
Malisa Jeanes
Joel Rueter
Tresa Sinha
Danika Loeffler
Chad Rahe
Joaquina Arbuckle
```

10.



```
[37] def print_collection_data():
       try:
         cursor = collections_1.find()
         for document in cursor:
           print(document)
       except Exception as e:
         print("An error occurred:", e)

     print_collection_data()
```

```
{'_id': 142, 'name': 'Laureen Salomone', 'scores': [{'score': 42.54322973844196, 'type': 'exam'}, {'score': 33.03152379449381, 'type': 'quiz'}, {'score': 77.52357320933667, 'type': 'homework'}]}
{'_id': 143, 'name': 'Gwyneth Garling', 'scores': [{'score': 44.29553481758053, 'type': 'exam'}, {'score': 23.15599504527296, 'type': 'quiz'}, {'score': 84.83695219376807, 'type': 'homework'}]}
{'_id': 144, 'name': 'Kaila Deibler', 'scores': [{'score': 20.85988856264308, 'type': 'exam'}, {'score': 73.51120532285645, 'type': 'quiz'}, {'score': 88.72483530139125, 'type': 'homework'}]}
{'_id': 145, 'name': 'Tandra Meadows', 'scores': [{'score': 19.07796402740767, 'type': 'exam'}, {'score': 7.63846325490759, 'type': 'quiz'}, {'score': 60.84655775785094, 'type': 'homework'}]}
{'_id': 146, 'name': 'Gwen Honig', 'scores': [{'score': 35.99646382910844, 'type': 'exam'}, {'score': 74.46323507534565, 'type': 'quiz'}, {'score': 90.95590422002779, 'type': 'homework'}]}
{'_id': 147, 'name': 'Sadie Jernigan', 'scores': [{'score': 6.14281392478545, 'type': 'exam'}, {'score': 44.94102013771302, 'type': 'quiz'}, {'score': 89.94407975401369, 'type': 'homework'}]}
{'_id': 148, 'name': 'Carli Belvins', 'scores': [{'score': 84.4361816750119, 'type': 'exam'}, {'score': 1.702113040528119, 'type': 'quiz'}, {'score': 88.48032660881387, 'type': 'homework'}]}
{'_id': 149, 'name': 'Synthia Labelle', 'scores': [{'score': 11.06312649271668, 'type': 'exam'}, {'score': 89.27462706564148, 'type': 'quiz'}, {'score': 41.1722010153017, 'type': 'homework'}]}
{'_id': 150, 'name': 'Eugene Magdaleno', 'scores': [{'score': 69.64543341032858, 'type': 'exam'}, {'score': 17.46202326917462, 'type': 'quiz'}, {'score': 39.41502498794787, 'type': 'homework'}]}
{'_id': 151, 'name': 'Meagan Oakes', 'scores': [{'score': 75.02808260234913, 'type': 'exam'}, {'score': 35.45524188731927, 'type': 'quiz'}, {'score': 75.84754202828454, 'type': 'homework'}]}
{'_id': 152, 'name': 'Richelle Siemers', 'scores': [{'score': 52.0158789874646, 'type': 'exam'}, {'score': 19.25549934746802, 'type': 'quiz'}, {'score': 68.33217408510437, 'type': 'homework'}]}
{'_id': 153, 'name': 'Mariette Batdorf', 'scores': [{'score': 91.38690728885123, 'type': 'exam'}, {'score': 39.98831767858929, 'type': 'quiz'}, {'score': 51.59702098442595, 'type': 'homework'}]}
{'_id': 154, 'name': 'Rachell Aman', 'scores': [{'score': 94.50988306850947, 'type': 'exam'}, {'score': 5.68414255121964, 'type': 'quiz'}, {'score': 64.46720717616572, 'type': 'homework'}]}
{'_id': 155, 'name': 'Aleida Elsass', 'scores': [{'score': 42.89558347656537, 'type': 'exam'}, {'score': 94.10647660402866, 'type': 'quiz'}, {'score': 30.56402201379193, 'type': 'homework'}]}
{'_id': 156, 'name': 'Kayce Kenyon', 'scores': [{'score': 54.00824880446614, 'type': 'exam'}, {'score': 19.20300722190935, 'type': 'quiz'}, {'score': 71.57649363606814, 'type': 'homework'}]}
{'_id': 157, 'name': 'Ernestine Macfarland', 'scores': [{'score': 9.666623747888858, 'type': 'exam'}, {'score': 98.76040135775126, 'type': 'quiz'}, {'score': 51.67453757397309, 'type': 'homework'}]}
{'_id': 158, 'name': 'Houston Valenti', 'scores': [{'score': 68.36209185504055, 'type': 'exam'}, {'score': 15.83819664395878, 'type': 'quiz'}, {'score': 81.7258704821604, 'type': 'homework'}]}
{'_id': 159, 'name': 'Terica Brugger', 'scores': [{'score': 97.822030541043, 'type': 'exam'}, {'score': 91.56280485763772, 'type': 'quiz'}, {'score': 62.01976292987356, 'type': 'homework'}]}
{'_id': 160, 'name': 'Lady Lefevers', 'scores': [{'score': 89.14702404133767, 'type': 'exam'}, {'score': 11.85715160788611, 'type': 'quiz'}, {'score': 87.70817474845785, 'type': 'homework'}]}
{'_id': 161, 'name': 'Kurtis Jiles', 'scores': [{'score': 38.84932631249875, 'type': 'exam'}, {'score': 75.6856190089661, 'type': 'quiz'}, {'score': 54.8262895255851, 'type': 'homework'}]}
{'_id': 162, 'name': 'Barbera Lippman', 'scores': [{'score': 10.1210778879972, 'type': 'exam'}, {'score': 57.39236107118298, 'type': 'quiz'}, {'score': 56.36039761834183, 'type': 'homework'}]}
{'_id': 163, 'name': 'Dinah Sauve', 'scores': [{'score': 9.660849614328693, 'type': 'exam'}, {'score': 0.710026283123355, 'type': 'quiz'}, {'score': 64.85706587155985, 'type': 'homework'}]}
{'_id': 164, 'name': 'Alica Pasley', 'scores': [{'score': 41.3852820348269, 'type': 'exam'}, {'score': 87.0183839032626, 'type': 'quiz'}, {'score': 37.22917544696978, 'type': 'homework'}]}
{'_id': 165, 'name': 'Elizabet Kleine', 'scores': [{'score': 23.35599596646158, 'type': 'exam'}, {'score': 45.42989961046475, 'type': 'quiz'}, {'score': 59.29421526983006, 'type': 'homework'}]}
{'_id': 166, 'name': 'Tawana Oberg', 'scores': [{'score': 79.24755285478162, 'type': 'exam'}, {'score': 97.28127199858804, 'type': 'quiz'}, {'score': 67.0528222080174, 'type': 'homework'}]}
{'_id': 167, 'name': 'Malisa Jeanes', 'scores': [{'score': 40.68676040665008, 'type': 'exam'}, {'score': 52.60826688242043, 'type': 'quiz'}, {'score': 94.67979508129564, 'type': 'homework'}]}
{'_id': 168, 'name': 'Joel Rueter', 'scores': [{'score': 21.78981361637835, 'type': 'exam'}, {'score': 1.182228345865832, 'type': 'quiz'}, {'score': 43.70843975739338, 'type': 'homework'}]}
{'_id': 169, 'name': 'Tresa Sinha', 'scores': [{'score': 52.22632020277269, 'type': 'exam'}, {'score': 65.68701091428014, 'type': 'quiz'}, {'score': 86.80410157346574, 'type': 'homework'}]}
{'_id': 170, 'name': 'Danika Loeffler', 'scores': [{'score': 80.13802901122058, 'type': 'exam'}, {'score': 9.613195588726075, 'type': 'quiz'}, {'score': 88.1580114788293, 'type': 'homework'}]}
{'_id': 171, 'name': 'Chad Rahe', 'scores': [{'score': 81.24054522370292, 'type': 'exam'}, {'score': 17.44929152365297, 'type': 'quiz'}, {'score': 82.77870021356301, 'type': 'homework'}]}
{'_id': 172, 'name': 'Joaquina Arbuckle', 'scores': [{'score': 35.43562368815135, 'type': 'exam'}, {'score': 89.74640983145014, 'type': 'quiz'}, {'score': 99.13868686848834, 'type': 'homework'}]}
{'_id': 173, 'name': 'Vinnie Auerbach', 'scores': [{'score': 57.26312067710243, 'type': 'exam'}, {'score': 20.63583040849144, 'type': 'quiz'}, {'score': 77.02638482252677, 'type': 'homework'}]}
```